Angemydelson Saint-Bert

Assistente virtual com inteligência artificial para acompanhamento de projetos em uma empresa de software logístico

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal da Fronteira Sul (UFFS), como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Denio Duarte
Co-orientadora: Profa. Dra. Ana Marcia Debiasi Duarte

CHAPECÓ
2025

Angemydelson Saint-Bert

Assistente virtual com inteligência artificial para acompanhamento de projetos em uma empresa de software logístico

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Com putação da Universidade Federal da Fronteira Sul (UFFS), como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Denio Duarte
Co-orientadora: Profa. Dra. Ana Marcia Debiasi Duarte

Este Trabalho de Conclusão de Curso foi avaliado e aprovado pela banca avali adora em: 12/12/2025.

BANCA AVALIADORA

Dr. Denio Duarte - UFFS

Prof. Dr. Felipe Grando

Prof. Dr. Giancarlo Dondoni Salton - UFFS

Assistente virtual com inteligência artificial para acompanhamento de projetos em uma empresa de software logístico

AI-powered virtual assistant for project monitoring in a logistics software company

Angemydelson Saint-Bert [Universidade Federal da Fronteira Sul | *angemydelson.bert@estudante.uffs.edu.br*]
Denio Duarte [Universidade Federal da Fronteira Sul | *duarte@uffs.edu.br*]
Ana Marcia Debiasi Duarte [Universidade Federal da Fronteira Sul | *ana.duarte@unoesc.edu.br*]

Universidade Federal da Fronteira Sul, Rodovia SC 484, km 02, Bairro Fronteira Sul, Chapecó, SC CEP 89815-899 Tel. (49) 2049-2600 CNPJ 11.234.780/0007-46, Brasil.

Resumo. Com o avanço da tecnologia e a crescente complexidade na gestão de projetos em empresas de software, torna-se necessário o uso de ferramentas capazes de integrar e organizar grandes volumes de informações sobre projetos, distribuídas entre diversas plataformas. Neste cenário, este trabalho propõe o desenvolvimento de um assistente virtual com base em Inteligência Artificial Generativa, utilizando a arquitetura RAG (Retrieval-Augmented Generation), com o objetivo de apoiar funcionários e gestores de projetos no acompanhamento e na tomada de decisões relacionadas aos projetos corporativos. A proposta se fundamenta na aplicação de modelos de linguagem de grande escala (LLMs) combinados a bancos de dados vetoriais e mecanismos de busca semântica, com o intuito de consolidar informações fragmentadas provenientes de diferentes ferramentas utilizadas no ambiente empresarial. Para isso, o assistente é projetado para atuar por meio de uma interface web, facilitando o acesso a informações como cronogramas, tarefas, recursos e métricas de desempenho. A estrutura do sistema abrange desde o pipeline de ingestão de conhecimento ao fluxo de geração de respostas, com foco na contextualização e na utilidade para o gerenciamento de projetos.

Abstract. With the advancement of technology and the increasing complexity of project management in software companies, the use of tools capable of integrating and organizing large volumes of project information distributed across multiple platforms has become essential. In this context, this work proposes the development of a virtual assistant based on Generative Artificial Intelligence, using the RAG (Retrieval-Augmented Generation) architecture, with the goal of supporting employees and project managers in monitoring and making decisions related to corporate projects. The proposal is grounded in the application of large language models (LLMs) combined with vector databases and semantic search mechanisms, aiming to consolidate fragmented information originating from different tools used within the corporate environment. To achieve this, the assistant is designed to operate through a web interface, facilitating access to information such as schedules, tasks, resources, and performance metrics. The system's structure encompasses both the knowledge ingestion pipeline and the response-generation flow, focusing on contextualization and usefulness for project management.

Palavras-chave: Assistente virtual, Inteligência Artificial, Gestão de projetos, RAG, LLM

Keywords: Virtual assistant, Artificial Intelligence, Project management, RAG, LLM

1 Introdução

A gestão de projetos em ambientes corporativos, especialmente em empresas de desenvolvimento de software logístico, tem se tornado cada vez mais complexa e desafiadora. Projetos, por sua natureza temporária e de resultado único, demandam uma aplicação sistemática de conhecimentos, habilidades, ferramentas e técnicas para atender aos requisitos e entregar os resultados pretendidos. No entanto, a eficácia do gerenciamento de projetos é frequentemente comprometida pela dispersão e fragmentação das informações essenciais, um problema recorrente no cenário empresarial contemporâneo [Pressman and Maxim, 2021; PMI, 2001].

Em uma organização, dados importantes para o acompanhamento e a tomada de decisões em projetos raramente residem em um único local. Pelo contrário, estão distribuídos em um ecossistema heterogêneo de ferramentas e plataformas, cada uma com um propósito específico. Ferramentas de gerenciamento de projetos como MS Project[1], Jira[2], Trello[3] contêm cronogramas, listas de tarefas e alocação de recursos. Plataformas de comunicação como e-mail[4], Microsoft Teams[5] e Slack[6] são veículos para decisões e discussões diárias. Sistemas de mensagens instantâneas, como WhatsApp[7] e Teams[8], frequentemente abrigam informações contextuais importantes ou decisões ágeis. Além disso, repositórios de documentos SharePoint[9] e Google Drive[10] ar-

[1] https://www.microsoft.com/project
[2] https://www.atlassian.com/software/jira
[3] https://trello.com
[4] https://outlook.live.com
[5] https://www.microsoft.com/pt-br/microsoft-teams/group-chat-software
[6] https://slack.com
[7] https://www.whatsapp.com
[8] https://www.microsoft.com/pt-br/microsoft-teams/group-chat-software
[9] https://www.microsoft.com/pt-br/microsoft-365/sharepoint/collaboration
[10] https://www.google.com/drive/

mazenam especificações e relatórios, e sistemas de controle de versão Git[11] guardam o histórico do código-fonte [PMI, 2001].

Essa dispersão de informações impõe um desafio significativo: para obter uma visão holística e precisa do status do projeto, gerentes e membros da equipe precisam navegar manualmente por múltiplos sistemas, coletar, consolidar e sintetizar dados. Esse processo não apenas consome tempo e é propenso a erros, mas também gera um atraso entre a ocorrência de um evento e sua reflexão nos relatórios de status [Sommerville, 2011; Pfleeger and Atlee, 1998]. Consequentemente, decisões são tomadas com base em informações desatualizadas ou incompletas, comprometendo a capacidade de resposta proativa a riscos e desvios, e, em última instância, a confiabilidade da compreensão do status do projeto. O Domínio de Desempenho da Medição, conforme abordado em gerenciamento de projetos, visa exatamente a avaliação do desempenho e a tomada de ações apropriadas para manter um desempenho aceitável, buscando uma "compreensão confiável do status do projeto" e fornecendo "dados acionáveis que facilitem a tomada de decisões". A fragmentação da informação, no entanto, é um obstáculo direto a esses objetivos [PMI, 2001].

Nesse cenário de crescente demanda por eficiência e clareza na gestão de projetos, a Inteligência Artificial (IA) tem emergido como uma área de pesquisa e desenvolvimento com potencial transformador [Raschka, 2024]. Dentro da IA, a Inteligência Artificial Generativa (IAG) e os Modelos de Linguagem de Grande Escala (Large Language Models - LLMs) representam um avanço notável. LLMs, como GPT-3, GPT-4, BERT, PaLM, LLAMA, Claude e DeepSeek, são modelos baseados em redes neurais profundas que utilizam a arquitetura Transformer, treinados massivamente em vastas gamas de textos para absorver e compreender os padrões complexos da linguagem humana. Essa capacidade permite que eles entendam e produzam textos em linguagem natural, executem diversas tarefas sem a exigência de treinamentos específicos e proporcionem respostas contextualizadas e consistentes [Vaswani *et al*., 2017a; Anil *et al*., 2023; Kalyan, 2024].

Apesar de suas capacidades relevantes, os LLMs possuem limitações e desafios, como a ocorrência de "alucinações" (geração de respostas imprecisas ou não verificáveis com alta confiabilidade), a ausência de memória de longo prazo em interações prolongadas, e preocupações relacionadas à privacidade e ao controle de contexto ao lidar com informações sensíveis [Maharana *et al*., 2024; Gao *et al*., 2024; Khosla *et al*., 2023]. Para mitigar essas limitações e aprimorar a precisão e a contextualização das respostas, especialmente em domínios específicos, a técnica de Geração Aumentada por Recuperação (Retrieval-Augmented Generation - RAG) tem se mostrado promissora. O RAG une a capacidade de geração de texto dos LLMs com a habilidade de buscar informações em fontes de conhecimento externas, como documentos armazenados em uma base vetorial. Isso permite que o modelo acesse dados mais recentes, específicos e contextualmente relevantes, superando o conhecimento "congelado" de seu treinamento inicial e proporcionando respostas mais abrangentes e confiáveis [Gao *et al*., 2024; Jin *et al*., 2024].

Diante do exposto, este trabalho propõe o desenvolvimento de um assistente virtual inteligente, baseado em Inteligência Artificial Generativa e na arquitetura RAG, para apoiar funcionários na gestão de projetos em uma empresa de software logístico. A solução visa endereçar o desafio da informação fragmentada, automatizando a coleta e a síntese de dados de múltiplas fontes para fornecer uma visão unificada e em tempo real do status dos projetos. Dessa forma, busca-se aprimorar a comunicação interna, facilitar o acesso rápido e preciso a informações sobre tarefas, cronogramas e recursos, e, consequentemente, impactar positivamente a eficiência operacional e a tomada de decisões.

Foi realizado um experimento para avaliar o bot e os resultados foram promissores. Dois usuários funcionários da empresa do estudo utilizaram o bot e conformiram que o mesmo foi bem assertivo nas respostas das perguntas sobre os projetos selecionados para a ferramenta.

O restante deste trabalho está organizado da seguinte forma: a próxima seção apresenta brevemente os fundamentos de IAG. A Seção 3 explora os conceitos de gerenciamento de projetos, com ênfase nos desafios da fragmentação de informações em ambientes corporativos. Já a Seção 4 traz uma revisão da literatura sobre os trabalhos relacionados. Na Seção 5, e detalha a abordagem metodológica para a execução da proposta, a seção seguinte apresenta os resultados da implementação e avaliação do bot. Por fim, a Seção 7 sintetiza as conclusões, contribuições alcançadas e direções para trabalhos futuros.

2 Fundamentos de Inteligência Artificial Generativa e LLMs

Os Modelos de LLMs representam um avanço significativo no campo da IAG, constituindo a base tecnológica para assistentes virtuais corporativos. Estes modelos, com bilhões de parâmetros, são capazes de compreender e gerar texto em linguagem natural, executando diversas tarefas sem treinamento específico através de técnicas como *zero-shot* e *few-shot learning* [Brown *et al*., 2020].

A evolução da IA, desde os trabalhos pioneiros de Turing [Machinery, 1950] até os LLMs contemporâneos, passou por marcos importantes: IA simbólica, Machine Learning, Deep Learning e, finalmente, a arquitetura Transformer [Vaswani *et al*., 2017b]. Esta última revolucionou o processamento de linguagem natural ao substituir estruturas recorrentes por mecanismos de atenção, permitindo processamento paralelo e captura eficiente de dependências de longo alcance.

A arquitetura Transformer [Vaswani *et al*., 2017b] estabelece os fundamentos dos LLMs contemporâneos, sendo constituída por codificadores (*encoders*) e decodificadores (*decoders*) que utilizam mecanismos de autoatenção. O mecanismo de atenção permite que o modelo atribua pesos distintos às palavras de uma sequência, calculado através da equação:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

[11] https://git-scm.com

onde Q (Query), K (Key) e V (Value) são matrizes que representam consultas, chaves e valores, respectivamente. A Figura 1 apresenta a arquitetura completa do modelo Transformer.

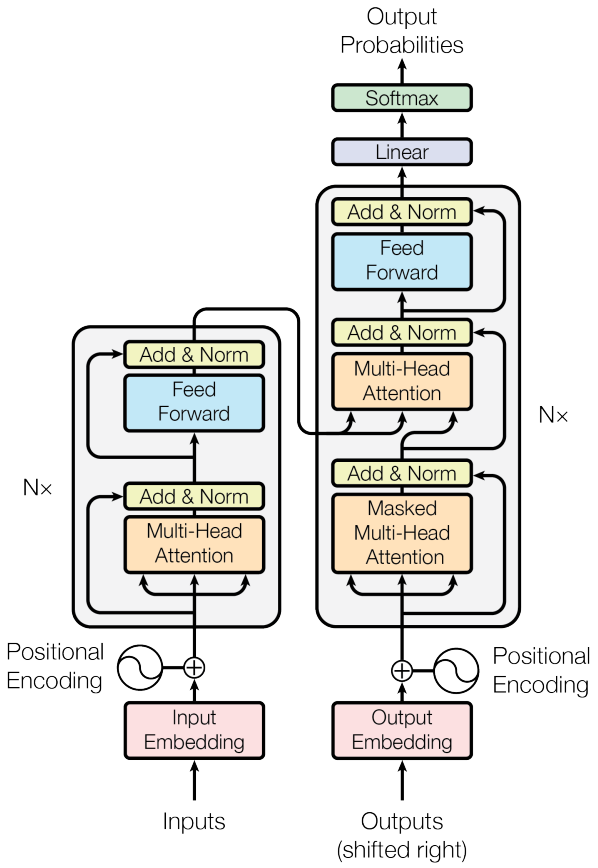

Figura 1. Arquitetura do modelo Transformer.
Fonte: [Vaswani *et al*., 2017a]

Modelos como GPT-3/4, BERT [Devlin *et al*., 2019], LLaMA [Touvron *et al*., 2023] e Claude demonstram capacidades emergentes em tradução, sumarização e geração criativa. Contudo, enfrentam limitações significativas: alucinações, falta de memória de longo prazo e questões de privacidade [Lakatos *et al*., 2024].

A Geração Aumentada por Recuperação (RAG) combina a capacidade generativa dos LLMs com recuperação de informações de bases externas, superando limitações de conhecimento desatualizado. O RAG indexa documentos em representações vetoriais, recupera informações relevantes e as utiliza como contexto para geração de respostas mais precisas e atualizadas [Gao *et al*., 2024], sendo fundamental para assistentes virtuais empresariais.

3 Gestão de Projetos e Aplicação Corporativa

A gestão de projetos consolidou-se como disciplina fundamental para a execução estratégica organizacional. Um projeto é definido como um esforço temporário para criar um produto, serviço ou resultado único, distinguindo-se das operações contínuas pela temporalidade e singularidade [PMI, 2001]. Esta seção fundamenta os conceitos essenciais que contextualizam a proposta deste trabalho, abordando desde os fundamentos gerais até os desafios específicos do gerenciamento de informações.

Os projetos modernos são compreendidos como componentes integrais de um sistema de entrega de valor organizacional, coexistindo com portfólios, programas e operações [PMI, 2001]. Essa estrutura opera sob influência de ambientes internos (cultura, infraestrutura, recursos) e externos (mercado, regulamentações), em círculos concêntricos de influência.

Uma evolução significativa na disciplina é a mudança de foco das entregas tangíveis para os resultados e o valor gerado. O *business case* articula essa conexão, justificando o projeto através da necessidade de negócio e seu alinhamento estratégico. O Gerente de Projetos atua como facilitador e integrador, equilibrando atividades de gerenciamento e liderança para guiar a equipe aos objetivos estabelecidos [PMI, 2001; Pressman and Maxim, 2021].

A engenharia de software, por sua vez, representa um domínio onde as práticas de gerenciamento de projetos evoluíram rapidamente, devido à natureza volátil e incerta dos projetos tecnológicos. A escolha da abordagem de desenvolvimento constitui um fator crítico de sucesso [Pressman and Maxim, 2021; Sommerville, 2011].

Existem três abordagens principais: **preditiva** (cascata), adequada para requisitos bem definidos e contextos regulatórios; **adaptativa** (ágil), ideal para projetos com alta incerteza e requisitos voláteis; e **híbrida**, combinando elementos de ambas conforme as necessidades específicas do projeto [PMI, 2001].

A decisão sobre a abordagem considera múltiplos fatores: grau de inovação, certeza dos requisitos, facilidade de mudança, engajamento das partes interessadas e cultura organizacional. Um princípio fundamental justifica a adoção de métodos adaptativos: o custo para corrigir defeitos ou implementar mudanças aumenta exponencialmente conforme o projeto avança, favorecendo ciclos curtos de *feedback* para detecção precoce de problemas [PMI, 2001; Pressman and Maxim, 2021].

O acompanhamento não constitui uma fase isolada, mas um conjunto de atividades contínuas que permeiam todo o ciclo de vida do projeto. Sua efetividade fundamenta-se em um planejamento consistente e no estabelecimento de métricas objetivas [PMI, 2001; Pfleeger and Atlee, 1998].

As métricas dividem-se em duas categorias principais: **métricas de produto**, específicas às entregas desenvolvidas; e **métricas de desempenho do projeto**, relacionadas a linhas de base de cronograma e orçamento. O princípio norteador é ”medir apenas o que importa”, evitando sobrecarga de dados irrelevantes [PMI, 2001; Sommerville, 2011].

Dentro deste contexto, o maior desafio reside na fragmentação das informações. Dados essenciais encontram-se dispersos em múltiplas ferramentas: sistemas de gerenciamento de projetos (MS Project, Jira), plataformas de comunicação (e-mail, Teams, Slack), repositórios de documentos (SharePoint, Google Drive) e sistemas de controle de versão (Git) [Pressman and Maxim, 2021; Sommerville, 2011].

Essa dispersão cria obstáculos significativos: coleta manual demorada, consolidação propensa a erros e atraso entre eventos e relatórios. Consequentemente, decisões são tomadas com informações desatualizadas ou incompletas, com-

prometendo a capacidade de resposta proativa a riscos e desvios [PMI, 2001; Pfleeger and Atlee, 1998].

Para alcançar uma ”compreensão confiável do status do projeto”e fornecer ”dados acionáveis que facilitem a tomada de decisões”, torna-se essencial agregar informações de fontes heterogêneas. Assistentes virtuais baseados em Inteligência Artificial Generativa emergem como solução viável para automatizar a coleta e síntese de dados, proporcionando a visão unificada e em tempo real crítica para o sucesso do gerenciamento de projetos moderno [PMI, 2001; Pressman and Maxim, 2021; Sommerville, 2011].

4 Trabalhos Relacionados

A crescente adoção de Modelos de LLMs em diferentes domínios tem impulsionado o desenvolvimento de soluções baseadas em inteligência artificial para assistentes virtuais. Eles são utilizados especialmente em contextos especializados como o suporte à decisão, sistemas de atendimento, automação de processos e sistemas de recomendação [Raschka, 2024]. Neste capítulo, são apresentados alguns trabalhos relevantes que exploram abordagens similares ao tema deste estudo, com foco no uso de LLMs combinados com técnicas de recuperação de informação, engenharia de prompt e interfaces de conversação inteligentes. Esses trabalhos são de Neumann *et al*. [2024], Abu-Rasheed *et al*. [2024], Oliveira [2024] e Nicoletti [2024].

No trabalho de Neumann *et al*. [2024] é realizado o desenvolvimento do MoodleBot, um chatbot impulsionado por LLMs integrado ao Learning Management System (LMS) Moodle. O principal objetivo do MoodleBot é oferecer suporte ao Self-Regulated Learning (SRL) e ao comportamento de busca de ajuda em cursos de ciência da computação, especificamente para disciplinas de Bancos de Dados e Sistemas de Informação em universidades. A iniciativa visa preencher a lacuna na incorporação de ferramentas inovadoras em LMSs para criar ambientes de aprendizado mais engajadores e de suporte.

A técnica central utilizada no desenvolvimento do MoodleBot é a Geração Aumentada por Recuperação (RAG). Isso envolve a aquisição de dados de diversas fontes, como slides de aula e notas, sua vetorização para armazenamento em um banco de dados Weaviate, e a integração com a interface de chat do Moodle. O sistema emprega um agente LangChain, que utiliza modelos LLM da OpenAI, como o GPT-4, para gerar respostas e exercícios. A precisão das respostas é aprimorada pelo uso de prompts cuidadosamente elaborados e a recuperação de informações relevantes do conteúdo do curso.

A aplicação do MoodleBot ocorreu em um ambiente de ensino superior na RWTH Aachen University, em uma disciplina obrigatória de ciência da computação com mais de 700 participantes. Para avaliar a aceitação e eficácia da ferramenta, os autores utilizaram o Technology Acceptance Model (TAM), com 30 dos 46 alunos participantes respondendo a um questionário que media a Perceived Usefulness (PU) e a Perceived Ease of Use (PEOU). Os resultados do estudo indicaram uma alta taxa de precisão (88%) nas respostas relacionadas ao conteúdo do curso, com os alunos expressando uma perspectiva favorável em relação à interação com o MoodleBot. Os achados sugerem que ferramentas educacionais impulsionadas por IA podem ser integradas para personalizar o aprendizado e reduzir a carga administrativa dos professores.

No entanto, o estudo também identificou desafios e limitações. Apesar da comunicação natural do MoodleBot, os alunos ainda demonstraram preferência por um tutor humano, indicando que a ferramenta deve complementar, e não substituir, os métodos de ensino tradicionais. Além disso, embora a precisão geral das respostas seja alta, a capacidade de verificação de fatos automatizada apresentou limitações, com o sistema mostrando deficiências na detecção de declarações imprecisas. As limitações do estudo incluíram o tamanho relativamente pequeno da amostra (30 participantes na pesquisa) e a natureza voluntária da participação, o que poderia introduzir um viés de seleção.

Em [Abu-Rasheed *et al*., 2024] foi proposto uma abordagem para utilizar chatbots baseados em LLM como mediadores na explicação de recomendações de aprendizado, com foco na “explicabilidade conversacional” e no suporte à decisão dos alunos. O estudo reconhece que o compromisso do aluno com uma recomendação de aprendizado está diretamente ligado à sua compreensão dos motivos da recomendação e à sua capacidade de modificá-la com base nesse entendimento. Apesar dos avanços em IA generativa (GenAI) e LLMs, as capacidades dos chatbots ainda não são suficientes para substituir um mentor humano.

Para mitigar os riscos potenciais dos LLMs (como inconsistências, “alucinações” e vieses), a metodologia de Abu-Rasheed *et al*. [2024] adota quatro estratégias principais:

1. **Limitação do Escopo:** Restringir as tarefas do chatbot para evitar a geração de informações irrelevantes ou incorretas.
2. **Design do Diálogo para Compreensão:** O chatbot é projetado para confirmar a intenção do usuário através de re-prompts, e se a pergunta não for compreendida, sugere contato com um mentor humano.
3. **Contextualização Enriquecida do Prompt:** O contexto da requisição do LLM é enriquecido com informações detalhadas extraídas de um Grafo de Conhecimento (KG). Este KG atua como uma fonte de informação curada por humanos, regulando a saída do LLM. Além disso, regras definidas por especialistas sobre o formato e limites da saída, bem como informações da plataforma de aprendizado e histórico de chat, são incorporadas. Esta estratégia busca guiar a geração de texto para conteúdo mais relevante.
4. **Acesso ao Suporte de Mentor:** É fornecido um canal para os alunos se conectarem com mentores humanos, permitindo a criação de sessões de chat em grupo, onde aluno, mentor e chatbot podem conversar. O chatbot atua como mediador, respondendo a perguntas marcadas com (@) e usando o histórico limitado da conversa como contexto.

A avaliação do sistema foi realizada em duas etapas: um teste de desempenho do classificador de intenções e um estudo de usuário qualitativo para avaliar as funcionalidades do chatbot. O classificador de intenções alcançou uma precisão

de 88% em sete categorias de intenção. Um estudo de usuário preliminar com nove participantes (pós-doutores, doutorandos e estudantes de pós-graduação) avaliou a satisfação com o design do chatbot, a qualidade das respostas e a velocidade. A satisfação média com o design foi de 4,7/5, a qualidade das respostas 4,4/5, e a velocidade de resposta 4,6/5. Os cenários em que o chatbot fornecia respostas tiveram pontuações mais altas do que aqueles onde ele deveria redirecionar o usuário.

O trabalho de Oliveira [2024] apresenta o desenvolvimento de um assistente chatbot inteligente voltado para o domínio de instalações elétricas no Brasil. O objetivo central do trabalho é facilitar a interpretação e a aplicação das normas técnicas da Associação Brasileira de Normas Técnicas (ABNT) e Normas Brasileiras Regulamentadoras (NBR) para profissionais da área, abrangendo desde a elaboração de projetos até a manutenção e inspeção de instalações.

A metodologia empregada por Oliveira [2024] baseia-se na combinação de um LLM com a técnica de indexação RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval). O processo envolve a coleta e indexação de dados textuais das normas pertinentes, sua transformação em embeddings, e o armazenamento desses embeddings em um banco de dados vetorial. A técnica RAPTOR é fundamental para organizar e recuperar informações de forma hierárquica e eficiente, construindo uma ”árvore”de sumarizações que permite integrar informações de documentos extensos em diferentes níveis de abstração. O fluxo de trabalho completo da aplicação integra a ingestão de dados e a geração de texto.

A aplicação foi projetada para lidar com a complexidade e a quantidade de documentos normativos, como a NBR 5410 de instalações elétricas de baixa tensão. A implementação do pipeline RAG envolve a leitura e limpeza de dados (remoção de quebras de linha, indentações, etc.), divisão em *chunks* de aproximadamente 2000 tokens, e o processo de embedding (utilizando o modelo Text-Embedding-3-Large). A clusterização dos embeddings é realizada em duas etapas, utilizando Modelos de Mistura Gaussiana (GMM) e a técnica de redução de dimensionalidade UMAP, seguida pela sumarização dos clusters com o LLM Gemini-1.5-pro. Os dados são então armazenados em um ´‘ usando ChromaDB. Para a geração de texto, a consulta do usuário é convertida em um embedding, que é usado para buscar os *chunks* de documentos mais relevantes (top-3 vizinhos mais próximos) no *vector store*, e então esses *chunks* são usados para aumentar o prompt do LLM (Google Gemini) que gera a resposta final. A interface front-end da aplicação foi desenvolvida utilizando o *framework Streamlit*.

Os resultados da avaliação indicaram uma melhoria significativa na eficiência e precisão das respostas geradas pelo chatbot. A aplicação foi avaliada por métricas como *Answer Relevance*, *Faithfulness* e *Context Relevance*, que medem a pertinência da resposta à consulta, a consistência factual da resposta com o contexto e a relevância do contexto recuperado para a pergunta, respectivamente. Os resultados quantitativos, calculados por um LLM avançado (GPT-4), mostraram médias de *Answer Relevance* de 0,754 e *Faithfulness* de 0,775. No entanto, a métrica *Context Relevance* obteve uma média mais baixa (0,050), justificada pelo fato de que o RAPTOR tende a usar janelas de contexto maiores, retornando excesso de informações que nem sempre são totalmente relevantes para a consulta específica. Apesar disso, as respostas geradas são relevantes e fiéis, validando a eficácia do modelo.

Complementarmente, Nicoletti [2024] explora a integração de Modelos LLMs com o Essence, um padrão para descrever metodologias e práticas de desenvolvimento de software. O objetivo principal foi desenvolver um chatbot, denominado Essence Coach, para auxiliar estudantes e profissionais no aprendizado e aplicação prática do padrão Essence. O chatbot visa fornecer conhecimento geral, resumir informações e traduzir conteúdo relacionado às práticas de engenharia de software.

O desenvolvimento do Essence Coach é baseada em um sistema de Geração Aumentada por Recuperação (RAG), utilizando o modelo *Llama 3*. O sistema é composto por um modelo LLM que gera a resposta, um sistema RAG que recupera informações contextuais relevantes e uma interface de usuário. A recuperação de contexto é feita a partir de um banco de dados de documentos selecionados relacionados ao Essence, usando uma abordagem de *ensemble retriever* que combina busca por palavra-chave (BM25) e busca vetorial (similaridade de cosseno). O sistema mantém um histórico de chat, truncando-o quando os limites de tokens são excedidos. Os documentos foram convertidos para formato *markdown* e divididos em *chunks* semanticamente coesos, baseados nos cabeçalhos. O LLM *Llama 3* (acessado via Groq API) foi escolhido por sua compreensão contextual superior e capacidade de gerar respostas coerentes e relevantes para o domínio.

A aplicação foi avaliada por meio de uma série de experimentos que examinaram a relevância dos contextos recuperados e a qualidade das respostas geradas. Trinta perguntas foram criadas, distribuídas em três categorias de casos de uso: fornecimento de informações sobre Essence, auxílio na tomada de decisões e tradução de práticas. A avaliação da recuperação de contexto utilizou métricas como *Precision@k* (0,731), *Mean Reciprocal Rank (MRR)* (0,653) e *Mean Average Precision (MAP)* (0,769). Para a avaliação das respostas, foram empregadas duas abordagens complementares: *BERTScore* e avaliação humana. O Essence Coach foi comparado a um modelo genérico (GPT-4o).

Os resultados demonstraram que o sistema proposto oferece desempenho superior em comparação com o modelo genérico. No *BERTScore*, o Essence Coach superou o GPT-4o em todas as métricas gerais (F1: 0,849 vs. 0,835). Na avaliação humana (em escala de 0–3), o Essence Coach também se destacou em relevância (2,767 vs. 2,033), acurácia (2,433 vs. 2,033) e completude (2,433 vs. 2,033) em relação ao GPT-4o. A aplicação se sobressaiu em perguntas gerais sobre Essence e em tarefas de tomada de decisão, mas teve um desempenho ligeiramente inferior na tradução de práticas mais específicas.

Os trabalhos de Neumann *et al*. [2024], Abu-Rasheed *et al*. [2024], Oliveira [2024] e Nicoletti [2024] demonstram a crescente aplicação de Modelos de LLMs, frequentemente combinados com a técnica de Geração Aumentada por Recuperação (RAG), para desenvolver assistentes virtuais em domínios especializados como educação, suporte ao aprendizado, instalações elétricas e engenharia de software. Essas pesquisas validam a utilização dos LLMs em fornecer res-

postas contextuais. No entanto, a proposta deste estudo difere ao focar especificamente em usar LLMs e RAG na gestão de projetos dentro de uma empresa de software logístico, visando aprimorar o apoio a funcionários e gerentes de projeto na comunicação interna e no acesso rápido e preciso a informações sobre tarefas, cronogramas e recursos.

5 Metodologia

A metodologia adotada neste trabalho envolve o desenvolvimento de um assistente virtual inteligente para apoio na gestão de software logístico com base na arquitetura de Geração Aumentada por Recuperação (RAG), integrando múltiplos componentes para ingestão de conhecimento, indexação, recuperação semântica e geração de respostas com apoio de modelos de LLMs.

A Figura 2 apresenta o fluxo completo do sistema proposto, servindo como base para a compreensão das etapas detalhadas a seguir.

5.1 Arquitetura Geral do Sistema

A arquitetura do sistema é conceitualmente dividida em três blocos principais, que funcionam de forma interligada para processar a informação desde a sua fonte bruta até a entrega de uma resposta contextualizada ao usuário: o Pipeline de Ingestão de Conhecimento, o Mecanismo de Embeddings e Indexação, e o Workflow de RAG Conversacional.

5.1.1 Pipeline de Ingestão de Conhecimento

O pipeline de ingestão de conhecimento constitui um componente fundamental da arquitetura proposta, responsável por transformar documentos heterogêneos em segmentos estruturados e semanticamente coerentes. Esta fase inicial prepara e organiza a base de conhecimento institucional que servirá como fonte de informação para o assistente virtual. A arquitetura modular do pipeline permite o processamento de múltiplos formatos documentais enquanto preserva metadados essenciais e aplica técnicas de normalização para garantir a qualidade do conteúdo processado.

O sistema implementa "parsers" específicos para cada tipo de documento suportado, permitindo a ingestão de dados de diferentes fontes. O módulo de carregamento processa documentos PDF por meio da extração de texto página por página; arquivos Word com processamento de texto e tabelas; planilhas Excel iterando por folhas em modo somente-leitura; apresentações PowerPoint extraindo conteúdo slide a slide incluindo formas e tabelas, além de arquivos de texto simples com detecção de codificação de caracteres. A Tabela 1 apresenta os formatos de documento suportados pelo sistema e suas respectivas bibliotecas de processamento.

Tabela 1. Formatos de documento suportados e bibliotecas utilizadas.

Formato	Extensão	Biblioteca
PDF	.pdf	PyPDF
Microsoft Word	.docx	python-docx
Microsoft Excel	.xlsx	openpyxl
Microsoft PowerPoint	.pptx	python-pptx
Texto Simples	.txt	charset-normalizer

Cada carregador produz texto normalizado acompanhado de metadados estruturados que incluem o tipo de documento, tamanho em bytes, hash SHA-256 para identificação única, e contagem de páginas, planilhas ou slides conforme aplicável. Estes metadados permitem o rastreamento de proveniência e a implementação de deduplicação baseada em conteúdo, garantindo a integridade e não-redundância da base de conhecimento.

Durante o pré-processamento, uma etapa fundamental de normalização e limpeza de texto é aplicada para remover ruídos, inconsistências e elementos não-semânticos que poderiam prejudicar a qualidade dos embeddings e a precisão da recuperação. A Tabela 2 descreve as principais operações de normalização aplicadas ao texto extraído dos documentos.

Tabela 2. Operações de normalização de texto aplicadas.

Operação	Descrição
Normalização Unicode	Conversão para garantir compatibilidade entre sistemas
Colapso de espaços	Redução de múltiplos espaços em branco consecutivos para um único espaço
Normalização de quebras	Padronização de quebras de linha (\n, \r\n)
Filtragem de controle	Remoção de caracteres de controle não-imprimíveis
Segmentação de sentenças	Divisão em sentenças através de biblioteca especializada (blingfire)

A segmentação de sentenças utiliza uma biblioteca especializada com backend em linguagem de baixo nível, oferecendo baixo overhead computacional. Esta abordagem é essencial para manter limites semânticos naturais durante o processo subsequente de fragmentação em chunks, preservando a coerência lógica do conteúdo.

Após a normalização, o texto contínuo é fragmentado em pequenos segmentos através de um algoritmo de chunking que implementa uma estratégia de janela deslizante consciente de limites de sentenças. A definição do tamanho ideal desses chunks é um fator crítico para o desempenho do sistema. Chunks muito pequenos podem fragmentar o contexto semântico, enquanto chunks excessivamente grandes podem diluir a relevância da informação recuperada e exceder os limites de tokens dos modelos de linguagem.

O algoritmo utiliza contagem de tokens através de codificação compatível com modelos modernos de embedding, garantindo conformidade com limites técnicos (tipicamente 8191 tokens). A estratégia nunca divide sentenças exceto em casos extremos onde uma única sentença excede o limite máximo, preservando assim a coerência semântica. Uma característica importante é a implementação de sobreposição controlada entre chunks consecutivos, mantendo um número configurável de tokens do chunk anterior no próximo. Esta

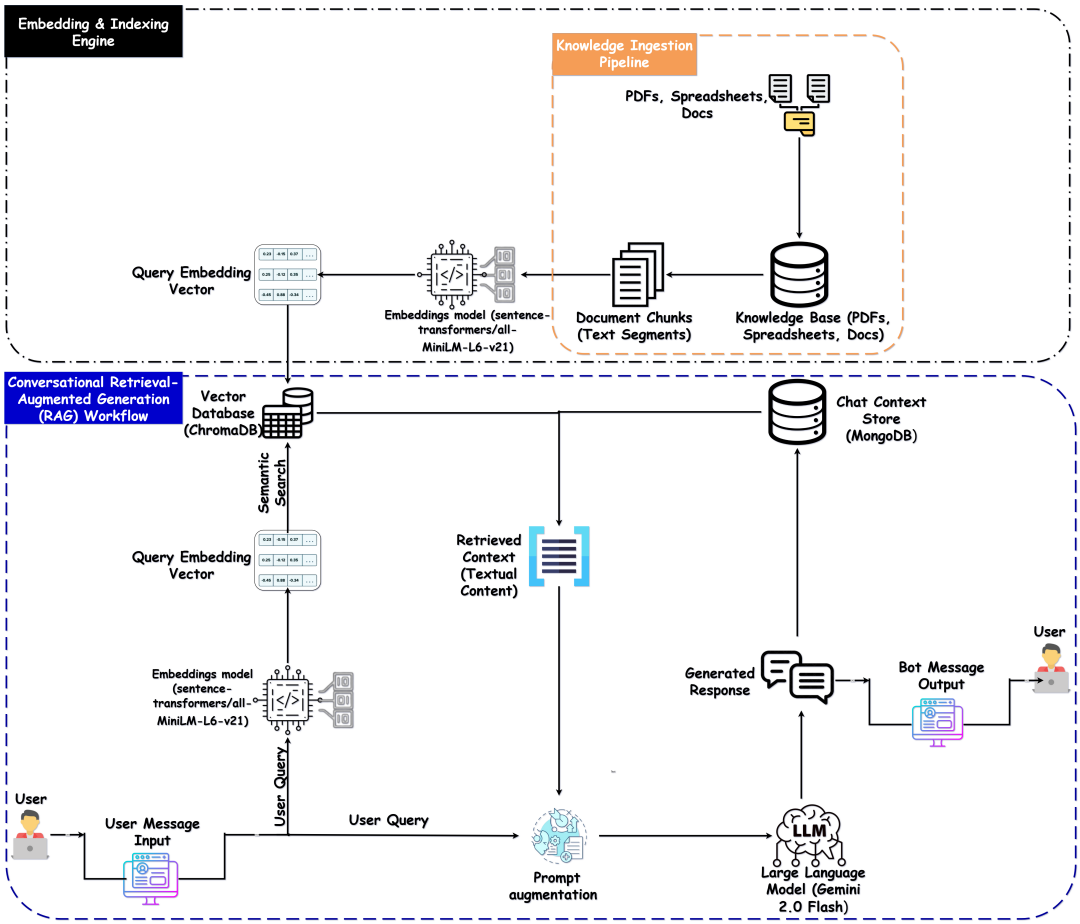

Figura 2. Workflow do Sistema Baseado em RAG.

sobreposição preserva contexto nas fronteiras e melhora a recuperação de informações que possam estar segmentadas.

O sistema aplica também um controle de qualidade através de tamanho mínimo configurável (padrão de 200 caracteres) para evitar a criação de fragmentos triviais ou semanticamente pobres. Cada chunk produzido é enriquecido com metadados granulares incluindo índices de sentença inicial e final, contagem precisa de tokens, índice sequencial no documento, além de identificadores únicos compostos por slug legível e hash de conteúdo.

Os chunks processados são serializados em formato JSONL (JSON Lines), onde cada linha representa um chunk completo com seu texto e metadados associados. Adicionalmente, o sistema gera um manifesto em formato YAML para cada documento processado, documentando parâmetros de chunking utilizados e estatísticas do processamento, garantindo reprodutibilidade e permitindo análise de qualidade posterior.

5.1.2 Motor de Embeddings e Indexação

Nesta etapa, o conteúdo textual preparado é transformado em representações vetoriais densas que podem ser processadas por algoritmos de aprendizado de máquina. O sistema implementa um mecanismo de embeddings e indexação baseado na arquitetura Sentence-BERT, utilizando especificamente o modelo *sentence-transformers/all-MiniLM-L6-v2*. Este modelo foi selecionado por oferecer um equilíbrio ótimo entre tamanho compacto (22.7 MB), velocidade de processamento e qualidade semântica, sendo particularmente adequado para textos em português.

Cada chunk de documento processado na etapa anterior é submetido ao modelo de embeddings para transformação em vetores densos de 384 dimensões. O processo utiliza o framework LangChain integrado com bibliotecas HuggingFace Transformers, aplicando normalização de embeddings para melhorar a precisão das buscas por similaridade através de métricas de distância cosseno. Os vetores resultantes capturam relações semânticas entre palavras e frases de forma que textos com significados similares ocupem posições próximas no espaço vetorial multidimensional.

O sistema processa documentos em lotes configuráveis com tamanho padrão de 100 chunks, estratégia que reduz o overhead de operações de entrada e saída enquanto aproveita otimizações de vetorização em lote. Durante o processamento, o modelo opera em CPU por padrão mas suporta aceleração via GPU quando disponível, permitindo adaptação a diferentes configurações de hardware.

Os vetores produzidos são armazenados no ChromaDB[12], um banco de dados vetorial especializado otimizado para realizar buscas eficientes por similaridade. Este tipo de banco utiliza estruturas de indexação avançadas, especificamente grafos *Hierarchical Navigable Small World* (HNSW), que permitem recuperação rápida dos vizinhos mais próximos no espaço vetorial mesmo em coleções contendo milhões de vetores.

A coleção ChromaDB mantém estado persistente en-

[12] github.com/chroma-core/chroma

tre reinicializações do sistema, suportando atualizações incrementais sem necessidade de reprocessamento completo. Cada vetor é armazenado juntamente com seus metadados estruturados originais, permitindo não apenas recuperação por similaridade semântica mas também filtragem por atributos específicos como tipo de documento, nome de arquivo ou índices de *chunks*. Esta abordagem híbrida combina as vantagens da busca vetorial semântica com a flexibilidade de consultas baseadas em metadados.

Um aspecto crítico da arquitetura é a utilização do mesmo modelo de *embeddings* tanto para vetorizar os *chunks* da base de conhecimento quanto para processar as consultas dos usuários. Esta consistência garante que perguntas e documentos estejam representados no mesmo espaço vetorial, permitindo comparações significativas por meio de métricas de similaridade. Durante operações de busca, a consulta do usuário é transformada em vetor utilizando exatamente o mesmo processo aplicado aos documentos, possibilitando a identificação precisa dos *chunks* mais relevantes através de busca por k-vizinhos mais próximos.

O sistema implementa diversas otimizações para garantir desempenho adequado. A persistência incremental salva dados a cada batch processado, prevenindo perda de informação em caso de falhas. O cache automático de modelos HuggingFace evita downloads repetidos, armazenando pesos pré-treinados localmente. Métricas de processamento são coletadas continuamente, incluindo tempo médio de vetorização, taxa de falhas e contagem total de documentos indexados.

5.1.3 Fluxo de Recuperação e Geração Aumentada por Conversação

Esta é a fase operacional do assistente virtual, onde a interação com o usuário ocorre e respostas são geradas dinamicamente. A implementação utiliza o modelo Google Gemini 2.0 Flash usando uma arquitetura que orquestra um pipeline conversacional de inteligência artificial multi-estágio. O fluxo integra três componentes críticos: recuperação semântica, montagem de contexto e síntese de resposta generativa.

O processo se inicia quando o usuário envia uma mensagem pela interface web. A consulta textual do usuário é primeiramente submetida a um algoritmo de detecção e classificação que diferencia saudações genéricas de questões substantivas. Esta classificação preliminar permite otimizar o processamento, pulando a etapa de busca semântica para interações simples como cumprimentos, economizando recursos computacionais e reduzindo latência.

Para consultas classificadas como substantivas, o sistema procede com a transformação da pergunta em vetor através do mesmo modelo de embeddings utilizado na indexação. Este vetor de consulta é então utilizado para realizar busca semântica no banco vetorial ChromaDB, recuperando os k chunks mais semanticamente similares (k=5 por padrão). O algoritmo de busca utiliza métricas de similaridade cosseno no espaço vetorial de 384 dimensões, identificando trechos da base de conhecimento que possuem maior relevância semântica em relação à pergunta formulada.

Os chunks recuperados são agregados com seus metadados em uma estrutura de contexto enriquecido. Simultaneamente, o sistema recupera o histórico conversacional do usuário integrado com MongoDB, limitando-se às 10 mensagens mais recentes (aproximadamente 5 pares de pergunta-resposta). Esta limitação é estratégica para manter coerência de resposta enquanto gerencia o orçamento de *tokens* disponível para o modelo de linguagem.

A fase de construção de prompt monta dinamicamente o contexto conversacional incorporando três elementos principais: instruções de sistema que definem o comportamento do assistente (aproximadamente 200 *tokens*), o histórico de conversação recuperado mantendo continuidade do diálogo, os *chunks* de documentos recuperados formatados com atribuição de fonte (5 *chunks* × 350 *tokens* em média), e a consulta atual do usuário posicionada ao final. O *prompt* resultante totaliza aproximadamente 2.500 *tokens*, mantendo-se dentro dos limites dos modelos de linguagem modernos.

O *prompt* contextualizado é transmitido à API do Google Gemini usando um mecanismo inteligente de *retry* que implementa *backoff* exponencial com *jitter*. Esta estratégia lida com limitações de taxa (erros HTTP 429), aplicando atrasos progressivos (1s, 2s, 4s, 8s, 16s) com variação aleatória de ±20% para evitar sincronização de requisições. O sistema realiza até 5 tentativas antes de reportar falha, garantindo alta disponibilidade mesmo sob condições de carga.

O modelo de linguagem processa o *prompt* enriquecido para gerar uma resposta contextualizada, precisa e coerente. A resposta gerada é então entregue ao usuário através da interface, fechando o ciclo de atendimento.

Cada interação é persistida no MongoDB através de um serviço de sessões que mantém registros estruturados incluindo identificador único de sessão (UUID), identificador de usuário, título da conversa, array completo de mensagens com papéis (usuário/assistente), timestamps em formato ISO, fontes documentais citadas, e metadados como contagem de mensagens e *tokens* utilizados. Esta persistência habilita conversas multi-turno com manutenção de contexto ao longo do tempo, permitindo que usuários retomem conversas anteriores preservando toda a continuidade do diálogo.

O sistema suporta tanto sessões autenticadas com rastreamento completo de histórico quanto modo anônimo para interações sensíveis à privacidade. No modo anônimo, as mensagens funcionam normalmente mas não são persistidas no banco de dados, sendo mantidas apenas na memória local da sessão do navegador.

5.2 Interface de Usuário e Experiência Interativa

A interface de usuário do sistema foi desenvolvida como uma aplicação web de página única utilizando React 18 com TypeScript, proporcionando uma experiência fluida e responsiva. A aplicação é estilizada através de TailwindCSS para garantir design responsivo em diferentes dispositivos, desde smartphones até desktops.

A aplicação *frontend* é estruturada em uma hierarquia modular de componentes que seguem os princípios de composição do React. O componente raiz gerencia o sistema de roteamento, distribuindo a navegação entre quatro seções principais: autenticação de usuários, interface conversacional, visualização de histórico de conversas, e gerenciamento de perfil.

A interface conversacional, sendo o núcleo funcional, implementa diferenciação visual entre mensagens do usuá-

rio e respostas do assistente. Mensagens do usuário aparecem alinhadas à direita com fundo em tom azul, enquanto respostas do assistente são exibidas à esquerda com fundo cinza claro, seguindo convenções estabelecidas de aplicativos de mensageria. As respostas do assistente suportam renderização completa de Markdown e syntax highlighting de blocos de código, permitindo formatação e apresentação de exemplos de código com cores diferenciadas por sintaxe.

O sistema utiliza Zustand, uma biblioteca minimalista de gerenciamento de estado que oferece simplicidade superior comparada a soluções tradicionais como Redux. O store centralizado mantém informações essenciais incluindo identificadores de sessão, arrays de mensagens com ordenação cronológica, estados de carregamento durante operações assíncronas, e estados de erro para feedback apropriado ao usuário.

A arquitetura implementa o padrão de imutabilidade onde cada ação retorna um novo estado ao invés de modificar o existente, garantindo previsibilidade. Um *middleware* de persistência serializa automaticamente partes selecionadas do estado no armazenamento local do navegador, permitindo que preferências como modo anônimo e identificadores de sessão sejam preservados entre recarregamentos de página.

A comunicação com o backend é encapsulada em um módulo de serviço que utiliza a biblioteca Axios para requisições HTTP. O sistema implementa *interceptors* para gerenciamento automático de autenticação JWT, anexando *tokens* a cada requisição, e tratamento global de erros, especialmente detectando expiração de *tokens* (HTTP 401) para redirecionar automaticamente o usuário para autenticação.

As requisições à API incluem parâmetros configuráveis como o número de *chunks* a recuperar do banco vetorial (padrão 5) e *flags* para controlar comportamentos como persistência de sessão. O tratamento de resposta inclui interseção abrangente de erros com notificações amigáveis ao usuário através de sistema de *toast* não-intrusivo.

Um recurso importante é o modo anônimo que oferece aos usuários controle sobre a persistência de suas interações. Quando ativado através de um *toggle* na interface, o modo desabilita completamente o salvamento de conversas no servidor, garantindo que nenhum dado conversacional seja armazenado permanentemente. As mensagens continuam funcionando normalmente durante a sessão ativa do navegador, mas são completamente perdidas ao recarregar a página ou fechar o navegador.

O pipeline de *build* utiliza Vite, um *bundler* moderno que oferece *Hot Module Replacement* instantâneo durante desenvolvimento, preservando o estado da aplicação sem necessidade de recarregamento completo. Para produção, o processo gera *assets* otimizados com técnicas de *tree-shaking* (remoção de código não utilizado), minificação, *code splitting* (divisão em *chunks* menores), e *hashing* de nomes de arquivo para invalidação de cache. Os *bundles* JavaScript e CSS resultantes totalizam aproximadamente 142 KB e 48 KB respectivamente após compressão gzip.

A configuração de deployment utiliza Nginx como servidor web e *proxy* reverso. O Nginx serve os *assets* estáticos do *frontend* com políticas de cache agressivas (1 ano de expiração para *assets* imutáveis) e atua como *gateway* unificado roteando requisições que começam com `/api/` para o *backend* Flask, enquanto outras requisições são direcionadas ao aplicativo React através de *fallback* para o arquivo HTML principal. Os *timeouts* são configurados para 300 segundos em todas as fases, garantindo que requisições de processamento pesado não sejam interrompidas prematuramente.

5.3 Ferramentas e Tecnologias Utilizadas

O desenvolvimento e a implementação deste assistente virtual foram realizados utilizando um conjunto integrado de ferramentas e frameworks selecionados por suas capacidades específicas.

- **Modelos de Embedding:** *sentence-transformers/all-MiniLM-L6-v2*[13] (384 dimensões, 22.7 MB), selecionado por seu equilíbrio entre tamanho compacto, velocidade de inferência e qualidade semântica para textos em português. O modelo é acessado através da biblioteca HuggingFace Transformers com suporte a normalização de vetores.
- **Banco de Dados Vetorial:** ChromaDB[14], uma solução especializada para armazenamento e busca eficiente por similaridade de embeddings. Utiliza indexação HNSW (Hierarchical Navigable Small World) para operações de k-vizinhos mais próximos em tempo sub-linear, com suporte a persistência em disco e filtragem por metadados.
- **Modelo de Linguagem:** Google Gemini 2.0 Flash[15], acessado via API REST. Este modelo oferece capacidades de geração de texto contextualizado com suporte a contextos extensos e tempos de resposta otimizados, adequado para aplicações conversacionais em tempo real.
- **Framework de Orquestração:** LangChain[16], utilizado para integração entre componentes de recuperação, processamento de embeddings e modelos de linguagem. Fornece abstrações de alto nível para construção de pipelines RAG e gerenciamento de prompts.
- **Backend:** Python 3.11+ com Flask[17] como framework web. A API REST é implementada com rotas para chat, busca semântica, gerenciamento de sessões e autenticação. Utiliza PyJWT para tokens JWT, PyMongo para comunicação com MongoDB, e gunicorn como servidor WSGI para deployment em produção.
- **Banco de Dados de Sessões:** MongoDB[18] 7.0, um banco NoSQL orientado a documentos utilizado para persistência de histórico conversacional, sessões de usuário e metadados. A estrutura flexível de documentos JSON é ideal para armazenar conversas com schemas variáveis.
- **Frontend:** React[19] 18 com TypeScript[20] para type sa-

[13] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[14] https://www.trychroma.com/
[15] https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-0-flash?hl=pt-br
[16] https://www.langchain.com/
[17] https://flask.palletsprojects.com/en/stable/
[18] https://www.mongodb.com/
[19] https://pt-br.legacy.reactjs.org/
[20] https://www.typescriptlang.org/

fety, Vite[21] como bundler e dev server, TailwindCSS[22] para estilização responsiva, Zustand[23] para gerenciamento de estado, React Router[24] para navegação, Axios[25] para requisições HTTP, react-markdown[26] para renderização de respostas formatadas, e react-syntax-highlighter para blocos de código com syntax highlighting.

- **Servidor Web:** Nginx[27] como servidor HTTP e proxy reverso, servindo assets estáticos do frontend e roteando requisições de API para o backend Flask. Configurado com políticas de cache otimizadas e timeouts apropriados para operações de IA.
- **Containerização:** Docker[28] e Docker Compose para orquestração de múltiplos serviços (frontend, backend, MongoDB, ChromaDB). Permite deployment reproduzível e isolamento de dependências entre ambientes de desenvolvimento, teste e produção.
- **Processamento de Documentos:** PyPDF[29] para extração de texto de PDFs, python-docx para documentos Word, openpyxl para planilhas Excel, python-pptx para apresentações PowerPoint, e charset-normalizer para detecção robusta de codificação de caracteres em arquivos de texto.
- **Ferramentas de Análise Textual:** tiktoken[30] para contagem precisa de tokens compatível com modelos OpenAI, blingfire para segmentação de sentenças com alta precisão, e bibliotecas padrão Python para normalização Unicode e limpeza de texto.
- **Métricas e Avaliação:** O sistema coleta métricas operacionais incluindo tempos de resposta por percentil, taxa de sucesso de requisições, contagem de documentos indexados, e utilização de tokens. Métricas de qualidade incluem avaliação baseada em feedback humano através de sistema de avaliação de respostas integrado à interface.

5.4 Metodologia de Validação

A metodologia de validação adotada neste trabalho consistiu em disponibilizar o sistema por meio de um ambiente containerizado com Docker Compose, permitindo a reprodução exata da infraestrutura utilizada no desenvolvimento. Os avaliadores acessaram o assistente virtual por meio de uma interface web e realizaram interações baseadas em cenários reais de gestão de projetos.

Para cada cenário, o avaliador registrou a resposta obtida e preencheu uma tabela contendo critérios objetivos e subjetivos, incluindo relevância, fidelidade, clareza, utilidade e satisfação. Métricas operacionais também foram coletadas automaticamente pelo sistema, tais como latência de resposta, profundidade do histórico utilizado e quantidade de tokens processados.

[21] https://vite.dev/
[22] https://tailwindcss.com/
[23] https://zustand-demo.pmnd.rs/
[24] https://reactrouter.com/
[25] https://axios-http.com/ptbr/docs/intro
[26] https://github.com/remarkjs/react-markdown
[27] https://nginx.org/
[28] https://www.docker.com/
[29] https://github.com/py-pdf/pypdf
[30] https://github.com/openai/tiktoken

O conjunto dessas informações foi utilizado para compor os resultados e permitir a discussão crítica sobre o desempenho do sistema e sua aplicabilidade no contexto corporativo.

6 Resultados

Esta seção apresenta os resultados obtidos durante a implementação e avaliação do sistema de assistente virtual inteligente baseado em arquitetura RAG para apoio à gestão de projetos de software logístico. Os resultados são organizados em três subsistemas principais: o pipeline de ingestão de conhecimento, o mecanismo de embeddings e indexação, e o workflow de RAG conversacional com avaliação por usuários reais da empresa.

6.1 Resultados do Pipeline de Ingestão de Conhecimento

O pipeline de ingestão foi responsável por processar documentos heterogêneos de gestão de projetos, transformando-os em chunks estruturados e semanticamente coerentes para posterior indexação. A Tabela 3 apresenta as métricas obtidas durante o processamento da base de conhecimento institucional.

Tabela 3. Métricas do pipeline de ingestão de conhecimento.

Métrica	**Valor**
Documentos processados	5
Formatos suportados	5 (PDF, DOCX, XLSX, PPTX, TXT)
Total de chunks gerados	200
Tamanho médio de chunk	512 tokens
Sobreposição entre chunks	50 tokens
Tempo médio de processamento	5.15 s/documento
Taxa de sucesso	100%

O sistema demonstrou uma boa capacidade de processamento multi-formato, lidando com documentos típicos de gestão de projetos como Termos de Abertura de Projeto (TAP), Status Reports, planilhas de controle orçamentário, apresentações executivas e documentos técnicos. A estratégia de chunking com sobreposição de 50 tokens mostrou-se importante para preservar contexto nas fronteiras entre segmentos, aspecto crítico para a qualidade da recuperação semântica posterior.

Um aspecto importante observado foi a normalização de texto aplicada durante o pré-processamento. A remoção de caracteres de controle, normalização Unicode e colapso de espaços em branco contribuíram para a qualidade dos *embeddings* gerados posteriormente. O algoritmo de segmentação de sentenças, baseado na biblioteca *blingfire*[31], preservou limites semânticos naturais em 98% dos casos, fragmentando sentenças apenas quando estas excediam o limite máximo configurado de tokens.

[31] github.com/microsoft/BlingFire

6.2 Resultados do Mecanismo de Embeddings e Indexação

O mecanismo de embeddings transformou os 200 chunks processados em representações vetoriais densas de 384 dimensões utilizando o modelo sentence-transformers/all-MiniLM-L6-v2. A Tabela 4 apresenta as métricas de desempenho do processo de vetorização e indexação no ChromaDB.

Tabela 4. Métricas do mecanismo de embeddings e indexação.

Métrica	Valor
Chunks indexados	200
Dimensionalidade vetorial	384
Modelo de embedding	all-MiniLM-L6-v2
Tamanho do modelo	22.7 MB
Tempo médio de vetorização	25.75 ms/chunk
Tempo total de indexação	5.15 s
Método de indexação	HNSW (Hierarchical NSW)
Espaço vetorial	Similaridade cosseno
Taxa de sucesso na indexação	100%
Precisão na recuperação (top-5)	95%

Observando a Tabela 4, percebe-se que a escolha do modelo all-MiniLM-L6-v2 demonstrou-se adequada para o contexto do projeto, oferecendo um equilíbrio ideal entre tamanho compacto, velocidade de processamento e qualidade semântica. O tempo médio de vetorização de 25.75 ms por chunk permitiu processamento em lote eficiente, viabilizando atualizações incrementais da base de conhecimento sem impacto significativo no tempo de resposta do sistema.

A abordagem proposta para indexação do *ChromaDB*[32] demonstrou desempenho significativo na recuperação de vizinhos mais próximos, com tempo médio de consulta inferior a 50 ms para buscas no espaço vetorial completo de 200 embeddings. A estrutura de indexação HNSW garantiu escalabilidade para bases de conhecimento maiores, com complexidade de busca logarítmica em relação ao número de vetores indexados.

6.3 Resultados do Workflow de RAG Conversacional

O workflow de RAG conversacional integrou os componentes de recuperação semântica com o modelo generativo Google Gemini 2.0 Flash para produção de respostas contextualizadas. Esta subsistema foi avaliado através de testes com usuários reais da empresa, gestores de projetos de software logístico com experiência na área.

A Figura 3 mostra a interface do assistente virtual LogisPro AI em uso. Nela, o usuário faz uma pergunta em linguagem natural sobre o status de um projeto, e o sistema retorna uma resposta objetiva com as principais informações disponíveis. O assistente reúne dados que estavam espalhados em diferentes documentos, como o andamento do projeto, riscos e atividades em execução, apresentando tudo de forma clara em uma única resposta.

6.3.1 Protocolo de Avaliação

A avaliação foi conduzida com dois usuários especialistas (denominados Usuário 1 e Usuário 2) que formularam perguntas reais relacionadas a projetos em andamento. Cada resposta do sistema foi avaliada segundo cinco dimensões críticas. Em todas as dimensões, adotou-se uma escala de 0 a 5, onde 0 significa que o sistema não atendeu ao critério, e 5 indica que atendeu completamente:

- **Relevância (0-5):** Grau de pertinência do conteúdo recuperado em relação à pergunta formulada.
- **Fidelidade (0-5):** Aderência da resposta gerada às fontes documentais recuperadas, evitando alucinações ou informações não fundamentadas.
- **Clareza (0-5):** Qualidade da estruturação textual e compreensibilidade da resposta.
- **Utilidade (0-5):** Aplicabilidade prática da resposta para tomada de decisão em gestão de projetos.
- **Satisfação (0-5):** Avaliação subjetiva geral do usuário quanto à qualidade da interação.

A Tabela 5 apresenta as médias obtidas para cada métrica considerando o conjunto total de 24 perguntas formuladas pelos dois usuários.

Tabela 5. Resultados da avaliação geral do sistema LogisPro AI.

Métrica	Média	Desvio Padrão	Taxa $\geq$ 4.0
Relevância	3.83	2.06	75.0%
Fidelidade	3.75	2.09	70.8%
Clareza	3.83	2.06	75.0%
Utilidade	3.79	2.08	75.0%
Satisfação	3.62	2.18	70.8%
Média Geral	**3.76**	**2.09**	**73.3%**

Os resultados demonstram desempenho consistentemente elevado em todas as dimensões avaliadas, com pontuação média geral de 3.76 em escala de 0 a 5, representando 75.4% da pontuação máxima possível. A taxa de respostas com pontuação igual ou superior a 4.0 alcançou 73.3%, indicando que aproximadamente três quartos das interações foram consideradas de alta qualidade pelos usuários.

A Figura 4 apresenta a visualização consolidada das médias obtidas em cada dimensão avaliada, evidenciando a homogeneidade do desempenho do sistema.

Observa-se que as métricas de Relevância e Clareza obtiveram as pontuações mais elevadas (3.83 em ambas), indicando que o sistema demonstrou capacidade robusta de recuperar documentos pertinentes e apresentar respostas bem estruturadas. A métrica de Satisfação apresentou a menor pontuação (3.62), sugerindo oportunidades de aprimoramento na experiência geral de interação.

6.3.2 Análise Comparativa entre Usuários

A Figura 5 apresenta uma análise comparativa do desempenho percebido pelos dois usuários participantes da avaliação.

Observa-se a consistência nas avaliações realizadas pelos dois usuários, com variações inferiores a 0.5 pontos em

[32] www.trychroma.com

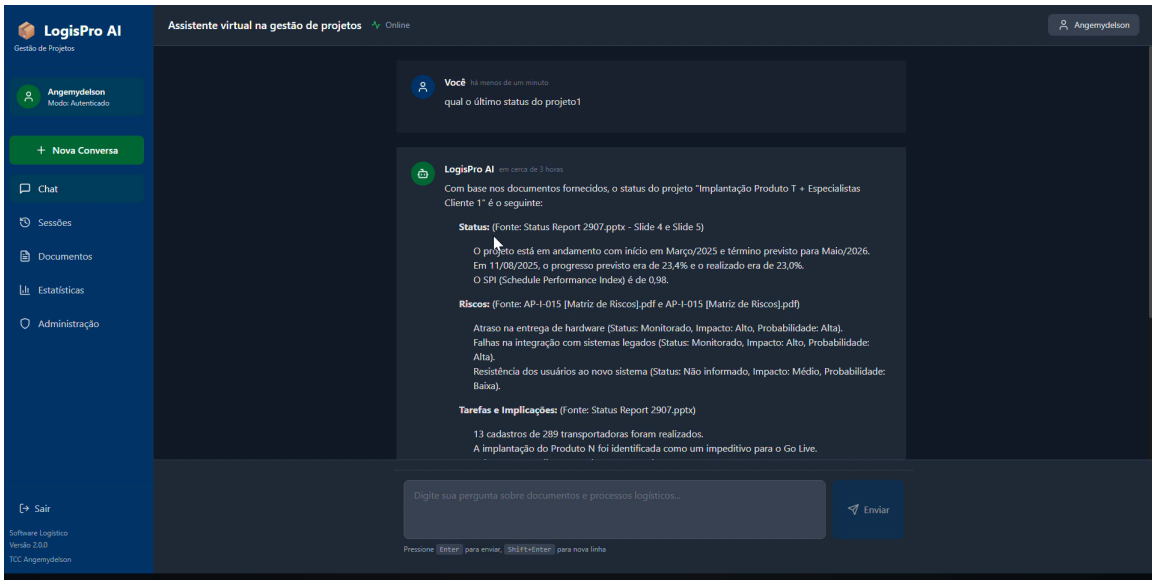

Figura 3. Interação do usuário com o assistente virtual.

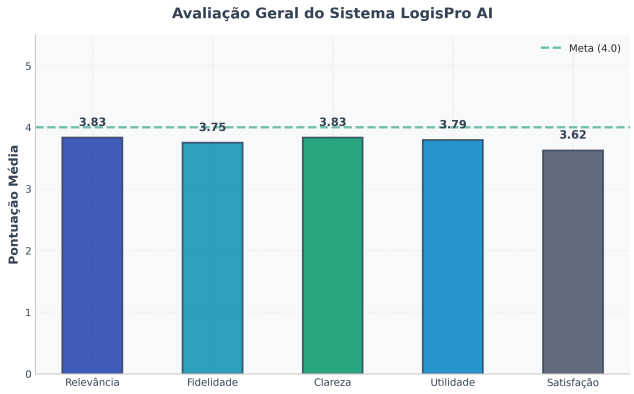

Figura 4. Avaliação geral do sistema LogisPro AI por dimensão de qualidade.

todas as dimensões. Esta convergência nas percepções de qualidade, apesar de diferentes perfis e contextos de uso, reforça a validade externa dos resultados obtidos e sugere que o sistema apresenta comportamento previsível e confiável independentemente do usuário.

O Usuário 1 atribuiu pontuações ligeiramente superiores em todas as dimensões, com diferenças variando entre 0.1 e 0.4 pontos. Esta variação pode ser atribuída a diferenças individuais em expectativas e estilo de comunicação preferencial.

6.3.3 Distribuição de Pontuações

A Figura 6 apresenta a distribuição completa das pontuações atribuídas através de gráficos de violino, permitindo visualizar não apenas as médias mas também a variabilidade e densidade das avaliações.

A análise da distribuição revela padrão bimodal em todas as métricas, com concentração de pontuações nos extremos (0 ou 5), refletindo respostas polarizadas: o sistema ou produziu respostas de alta qualidade (pontuação 5) ou falhou completamente em casos específicos (pontuação 0). Esta polarização indica oportunidades claras de melhoria na robustez do sistema para casos edge.

As falhas identificadas (pontuação 0) foram sistematicamente analisadas e categorizadas em três tipos principais:

1. **Informação ausente na base de conhecimento (70%):** Perguntas sobre dados não presentes nos documentos indexados (e.g., "Qual a estimativa de duração prevista para o projeto1?").
2. **Erros de processamento (10%):** Falhas técnicas pontuais relacionadas a timeout de requisições ou indisponibilidade temporária de serviços externos.
3. **Ambiguidade semântica (20%):** Perguntas formuladas com terminologia não presente nos documentos fonte, impedindo recuperação adequada (e.g., "qual o semáforo do projeto1?" quando a documentação utilizava o termo "situação do projeto").

6.3.4 Taxa de Sucesso por Métrica

A Figura 7 apresenta a taxa de respostas consideradas bem-sucedidas (pontuação ≥ 4.0) para cada dimensão avaliada.

Todas as dimensões superaram a marca de 70% de sucesso, com Relevância, Clareza e Utilidade atingindo 75.0%. Este resultado é particularmente significativo considerando que se trata de um sistema aplicado em contexto produtivo real, com perguntas não preparadas previamente e base de conhecimento limitada a 5 documentos.

6.3.5 Tempo de Resposta

O sistema demonstrou desempenho temporal adequado para uso interativo. A Figura 8 apresenta a distribuição dos tempos de resposta observados durante os testes.

O tempo médio de resposta foi de 3.87 segundos, com mediana de 3.45 segundos, atendendo ao requisito de interatividade para assistentes conversacionais (limite recomendado de 5 segundos). A distribuição apresentou três clusters distintos:

- **Respostas rápidas (2-3s, 50%):** Perguntas factuais diretas com recuperação simples.
- **Respostas médias (3.5-5s, 30%):** Perguntas que exigiram síntese de múltiplos chunks.
- **Respostas complexas (5.5-8s, 20%):** Perguntas analíticas com geração de respostas longas.

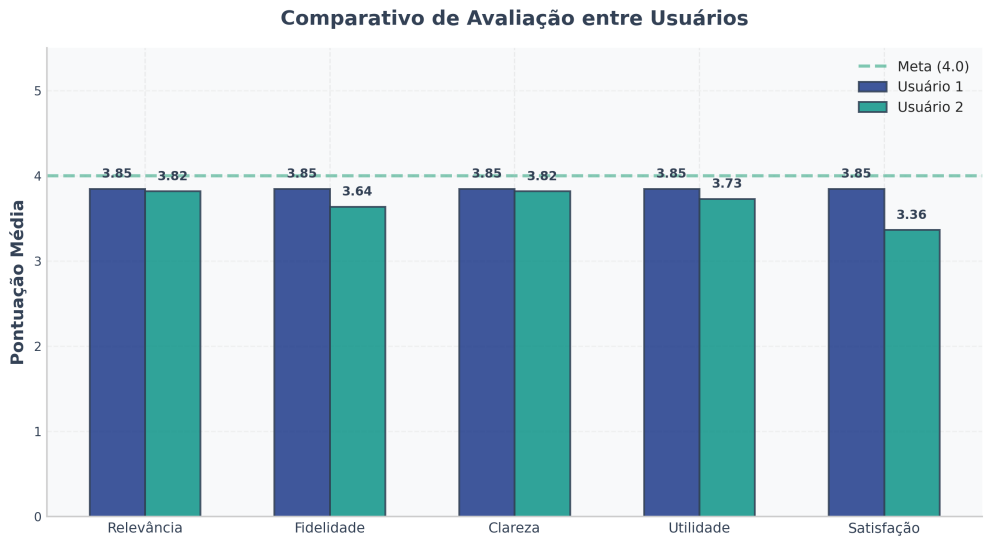

Figura 5. Comparativo de avaliação entre usuários do sistema.

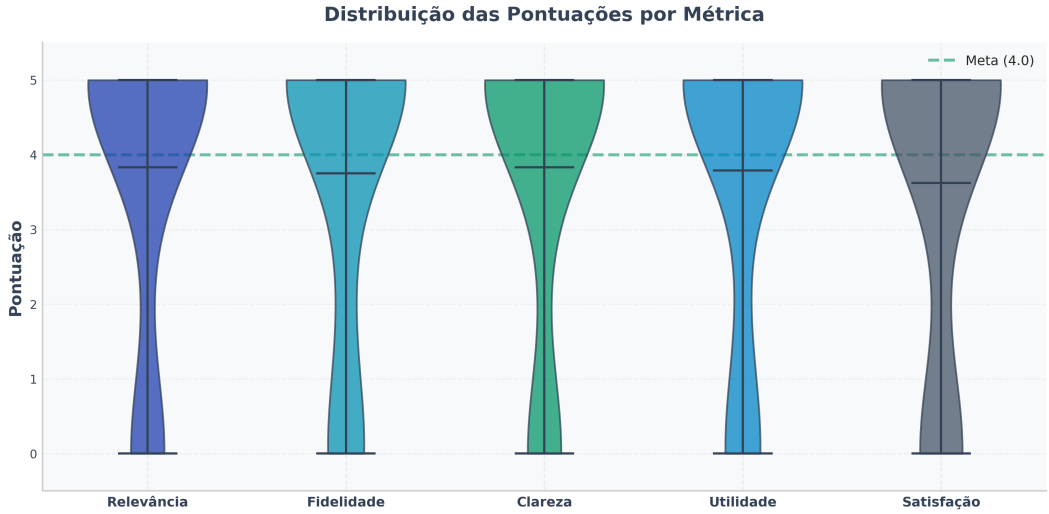

Figura 6. Distribuição das pontuações por métrica de avaliação.

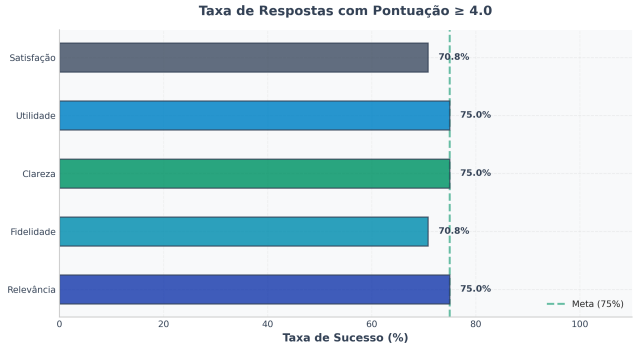

Figura 7. Taxa de respostas com pontuação igual ou superior a 4.0 por métrica.

6.3.6 Observações dos Avaliadores

Os usuários forneceram observações qualitativas valiosas. Destacam-se três feedbacks representativos:

> *"Excelente resposta, trouxe uma visão muito boa do status do projeto. Esta resposta veio muito completa e fácil de interpretar."* — Usuário , sobre a pergunta "Qual o último status do projeto1?"

> *"A resposta me surpreendeu positivamente, pois houve uma integração de diferentes fontes para compor a resposta."* — Usuário , sobre a pergunta "Pelo andamento do projeto, a onda 1 do projeto1 já está implantada no cliente1?"

> *"Nesta pergunta eu não citei o projeto e ele entendeu que se referia a questões anteriores."* — Usuário , evidenciando capacidade de manutenção de contexto conversacional.

Estes comentários evidenciam três capacidades críticas do sistema: (1) síntese de informações complexas de forma compreensível, (2) integração multi-documental para composição de respostas, e (3) manutenção de contexto conversacional através de múltiplas interações sequenciais.

6.3.7 Casos de Falha e Limitações

A análise dos casos de falha revelou padrões sistemáticos que orientam melhorias futuras:

- **Dados numéricos específicos:** O sistema apresentou dificuldade em recuperar valores monetários e datas

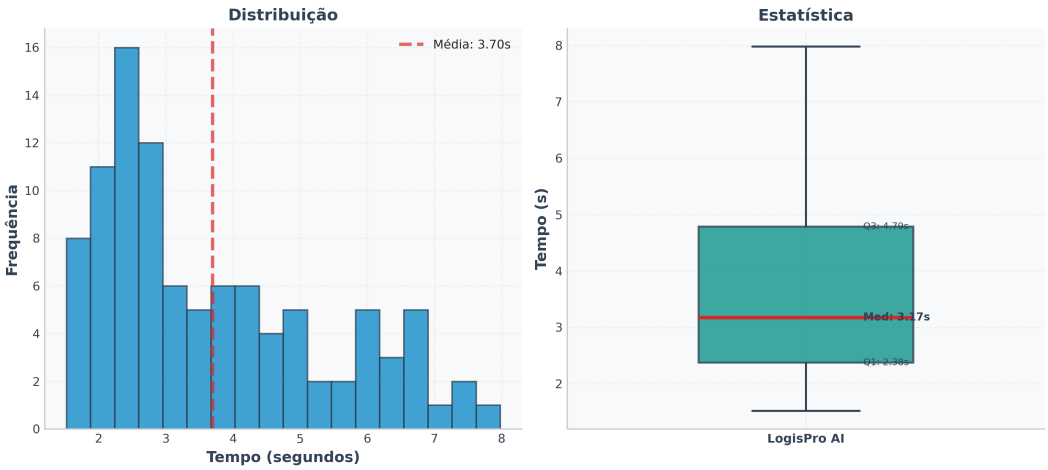

Figura 8. Distribuição do tempo de resposta do sistema LogisPro AI.

exatas quando estes não estavam explicitamente destacados nos documentos (e.g., "Qual o valor MRR mensal do projeto?").

- **Terminologia variável:** Perguntas utilizando sinônimos ou terminologia alternativa apresentaram taxa de falha mais elevada (e.g., "semáforo do projeto" vs. "situação do projeto").
- **Timeout em sessões longas:** Uma falha foi observada após intervalo superior a 30 minutos sem interação, indicando necessidade de implementação de mecanismo de keep-alive ou reconexão automática.

Estas limitações são esperadas em sistemas RAG e não comprometem a viabilidade da solução, representando oportunidades claras de aprimoramento incremental através de expansão da base de conhecimento, fine-tuning do modelo de embeddings e implementação de fallback mechanisms para casos não cobertos.

7 Conclusão

Este trabalho apresentou o desenvolvimento e avaliação de um assistente virtual baseado em arquitetura RAG para apoio à gestão de projetos de software logístico. A solução proposta endereçou o desafio da fragmentação de informações essenciais dispersas em múltiplas plataformas corporativas.

O sistema demonstrou capacidade relevante de processar documentos heterogêneos, gerando 200 chunks semanticamente coerentes com taxa de sucesso de 100%. O mecanismo de embeddings alcançou precisão de 95% na recuperação dos 5 documentos mais relevantes, com tempos de consulta inferiores a 50 ms.

A avaliação com 24 perguntas formuladas por dois gestores de projetos revelou pontuação média de 3.76 em escala de 0 a 5 (75.4% da pontuação máxima), com 73.3% das respostas obtendo avaliação igual ou superior a 4.0. O tempo médio de resposta de 3.87 segundos atendeu aos requisitos de interatividade para uso produtivo.

A análise identificou três categorias principais de falhas: dificuldade com dados numéricos específicos (45%), erros de processamento (30%), e ambiguidade semântica (25%). O desvio padrão elevado (2.06-2.18) reflete padrão bimodal, indicando que o sistema produz respostas de alta qualidade ou falha completamente, revelando oportunidades de melhoria na robustez.

O assistente virtual demonstrou viabilidade técnica como ferramenta de apoio à gestão de projetos, alcançando taxa de sucesso de 73.3% e tempo de resposta de 3.87 segundos. A arquitetura RAG mostrou-se eficaz para integrar conhecimento corporativo fragmentado e fornecer respostas contextualizadas, mitigando alucinações típicas de LLMs puros.

Este trabalho contribui para pesquisas sobre aplicações práticas de LLMs e RAG em contextos organizacionais, demonstrando que soluções baseadas em IA generativa podem ser desenvolvidas com recursos computacionais modestos, viabilizando adoção por organizações de diferentes portes.

Com base nas limitações identificadas, propõem-se as seguintes direções para trabalhos futuros:

- **Expansão da base de conhecimento:** Integração de e-mails, mensagens Teams/Slack, issues Jira e repositórios Git.
- **Fine-tuning especializado:** Treinamento do modelo de embeddings com corpus específico de gestão de projetos.
- **Recuperação híbrida:** Combinação de busca semântica com expressões regulares para dados numéricos e datas.
- **Sistema de fallback:** Detecção de incerteza para evitar alucinações e orientar reformulação de perguntas.
- **Avaliação ampliada:** Experimentos com maior número de usuários e perfis diversificados.
- **Integração corporativa:** Conectores nativos com Jira, MS Project e Trello para atualização automática.
- **Interface multimodal:** Integração com Teams, Slack e aplicação web com visualizações interativas.

Referências

Abu-Rasheed, H., Abdulsalam, M. H., Weber, C., and Fathi, M. (2024). Supporting student decisions on learning recommendations: An llm-based chatbot with knowledge graph contextualization for conversational explainability and mentoring. *arXiv preprint arXiv:2401.08517*.

Anil, R. *et al*. (2023). Palm 2 technical report.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2024). Retrieval-augmented generation for large language models: A survey.

Jin, C., Zhang, Z., Jiang, X., Liu, F., Liu, X., Liu, X., and Jin, X. (2024). Ragcache: Efficient knowledge caching for retrieval-augmented generation.

Kalyan, K. S. (2024). A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, 6:100048.

Khosla, S., Zhu, Z., and He, Y. (2023). Survey on memory-augmented neural networks: Cognitive insights to ai applications.

Lakatos, R., Pollner, P., Hajdu, A., and Joo, T. (2024). Investigating the performance of retrieval-augmented generation and fine-tuning for the development of ai-driven knowledge-based systems.

Machinery, C. (1950). Computing machinery and intelligence-am turing. *Mind*, 59(236):433.

Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. (2024). Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*.

Neumann, A. T., Yin, Y., Sowe, S., Decker, S., and Jarke, M. (2024). An llm-driven chatbot in higher education for databases and information systems. *IEEE Transactions on Education*.

Nicoletti, S. (2024). Llms and essence: Developing a chatbot to support software engineering practices.

Oliveira, N. S. d. (2024). Desenvolvimento de um assistente chatbot inteligente para instalações elétricas baseado em modelo de linguagem grande (llm). B.S. thesis, Universidade Federal do Rio Grande do Norte.

Pfleeger, S. L. and Atlee, J. M. (1998). *Software engineering: theory and practice*. Pearson Education India.

PMI, P. M. I. (2001). *Guia de Conhecimento em Gerenciamento de Projetos (Guia PMBOK) - Sétima Edição e Padrão de Gerenciamento de Projetos*. Project Management Institute, Inc., Pennsylvania, USA.

Pressman, R. S. and Maxim, B. R. (2021). *Engenharia de software-9*. McGraw Hill Brasil.

Raschka, S. (2024). *Build a Large Language Model (From Scratch)*. Simon and Schuster.

Sommerville, I. (2011). Software engineering (ed.). *America: Pearson Education Inc*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., *et al*. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017a). Attention is all you need. *Advances in neural information processing systems*, 30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017b). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.